What is claimed is:

1. A method for processing IP datagrams using an outbound processing state machine in an outbound processor, wherein the IP datagrams are generated by a host system, comprising:

creating an input/output control block ("IOCB") with plural host memory addresses that define host data to be sent and a host memory address of a network control block ("NCB") used to build network protocol headers, wherein the host sends the IOCB to the outbound processor.

2. The method of Claim 1, wherein the outbound processor reads the NCB from host memory and creates an IP and MAC level protocol header(s) for a data packet(s) used to send the IP data.

3. The method of Claim 1, wherein if a datagram fits into an IP packet, the outbound processor builds headers to send the datagram and then uses the plural host memory addresses defining the host data to read the data from the host, places the data into the packet and sends the packet.

4. The method of Claim 1, wherein if a datagram is greater than a certain size, the outbound processor generates packets with fragments of the datagram using the NCB information to build headers and then uses the plural host memory addresses defining the host data to read the data from the host, places the fragments of the datagram into each packet and sends the packets.

5. The method of Claim 4, wherein fragment flags indicate which particular fragment is being sent at any given time.

6. A method for processing TCP data packets generated by a host system using an outbound processing state machine in an outbound processor, comprising:

creating an input/output control block ("IOCB") with plural host memory addresses that define the host data to be sent and a memory address of a network control block ("NCB") used to build network protocol headers, wherein the host sends the IOCB to the outbound processor;

verifying if a TCP window is open;

building TCP/IP/MAC headers; and

sending the data packet(s).

7. The method of Claim 6, wherein the outbound processor reads the NCB into a local memory for creating the network protocol headers.

8. The method of Claim 6, wherein the outbound processor builds TCP headers which includes setting a source and destination port numbers, TCP sequence numbers, and other flags that indicate a type of packet, TCP header length and window size.

9. The method of Claim 6, wherein the outbound processor reads host data, appends the host data to a TCP header and calculates a TCP checksum while transferring the data.

10. The method of Claim 6, wherein the outbound processor inserts a calculated checksum into a previously built TCP header and then sends the packet.

11. The method of Claim 6, further comprising:

linking the NCB to a re-transmission timer list;

updating the NCB with a last sequence number of the data transmitted;

linking an original IOCB to the NCB, as a delayed request, in case all the data was not transmitted due to a window closing or if a re-transmission is necessary; and

storing the NCB waiting for an Acknowledgement.

12. A method for processing a TCP data transmit request after a TCP window is closed and then reopened by the reception of an acknowledgement (ACK) packet using an outbound processor, comprising:

reading a network control block (NCB) into a local memory;

reading a delayed request (IOCB) linked to the NCB;

verifying if a TCP window is open;

building TCP/IP/MAC headers; and

sending the data packet(s).

13. The method of Claim 12, wherein the outbound processor determines if a requested data transfer has been completed and generates an outbound TCP completion message.

14. A method for processing fragmented IP datagrams received from a network, comprising:

receiving the IP fragments into buffers in a local memory;

linking the IP fragment to a reassembly list for a particular IP datagram; and

when all fragments are present, sending the complete datagram to TCP or a host for additional processing.

15. The method of Claim 14, wherein if the IP fragment is the first fragment received for a datagram, a new reassembly list is created.

16. The method of Claim 15, wherein after a new reassembly list is created, a timer is started to ensure the reassembly is completed in a certain amount of time.

17. The method of Claim 14, wherein if the IP fragment is not the first fragment received for a datagram and is in order with the fragments already on the list, the fragment is added to the end of the list.

18. The method of Claim 14, wherein if the IP fragment is not the first fragment received for a datagram and is out of order with the fragments already on the list, the fragment is inserted into the reassembly list as indicated by its IP offset.